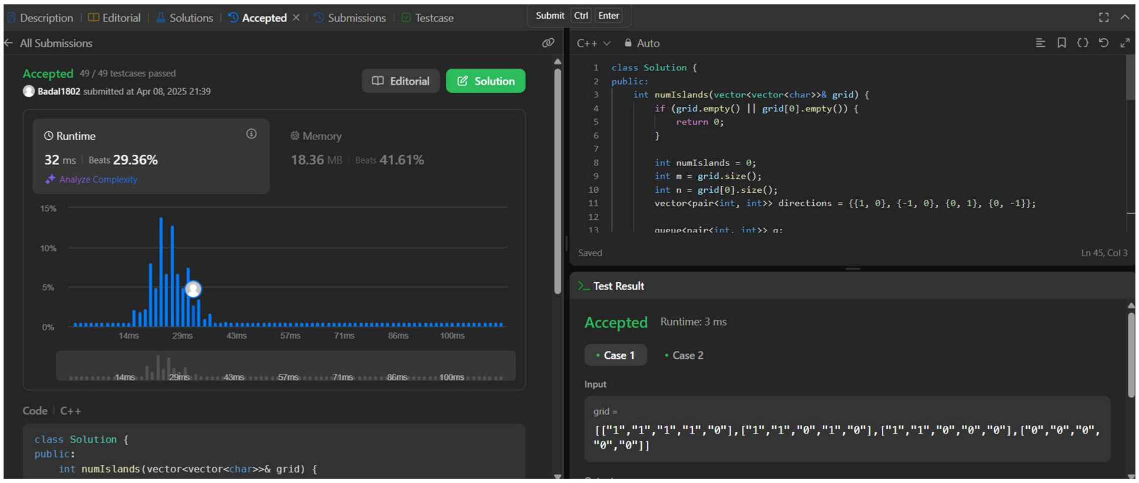
Experiment 9

Number of Islands

CODE:

```
class Solution {
public:
    int numIslands(vector<vector<char>>& grid) {
        if (grid.empty() || grid[0].empty()) {
            return 0;
        }

        int numIslands = 0;
        int m = grid.size();
        int n = grid[0].size();
        vector<pair<int, int>> directions = {{1, 0}, {-1, 0}, {0, 1}, {0, -1}};

        queue<pair<int, int>> q;

        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                if (grid[i][j] == '1') {
                    numIslands++;
                    q.push({i, j});

                    while (!q.empty()) {
                        auto [x, y] = q.front();
                        q.pop();

                        if (x < 0 || x >= m || y < 0 || y >= n || grid[x][y] != '1') {
                            continue;
                        }

                        grid[x][y] = '0'; // mark as visited

                        for (auto& dir : directions) {
                            int nx = x + dir.first;
                            int ny = y + dir.second;
                            if (nx >= 0 && nx < m && ny >= 0 && ny < n && grid[nx][ny] == '1') {
                                q.push({nx, ny});
                            }
                        }
                    }
                }
            }
        }

        return numIslands;
    }
};
```

OUTPUT:



Accepted 49 / 49 testcases passed

Badal1802 submitted at Apr 08, 2025 21:39

Runtime: 32 ms | Beats 29.36%

Memory: 18.36 MB | Beats 41.61%

```
class Solution {
public:
    int numIslands(vector<vector<char>>& grid) {
```

```
class Solution {
public:
    int numIslands(vector<vector<char>>& grid) {
        if (grid.empty() || grid[0].empty()) {
            return 0;
        }

        int numIslands = 0;
        int m = grid.size();
        int n = grid[0].size();
        vector<pair<int, int>> directions = {{1, 0}, {-1, 0}, {0, 1}, {0, -1}};

        queue<pair<int, int>> q;
```

Accepted Runtime: 3 ms

```
grid =
[["1","1","1","1","0"],["1","1","0","1","0"],["1","1","0","0","0"],["0","0","0",
"0","0"]]
```

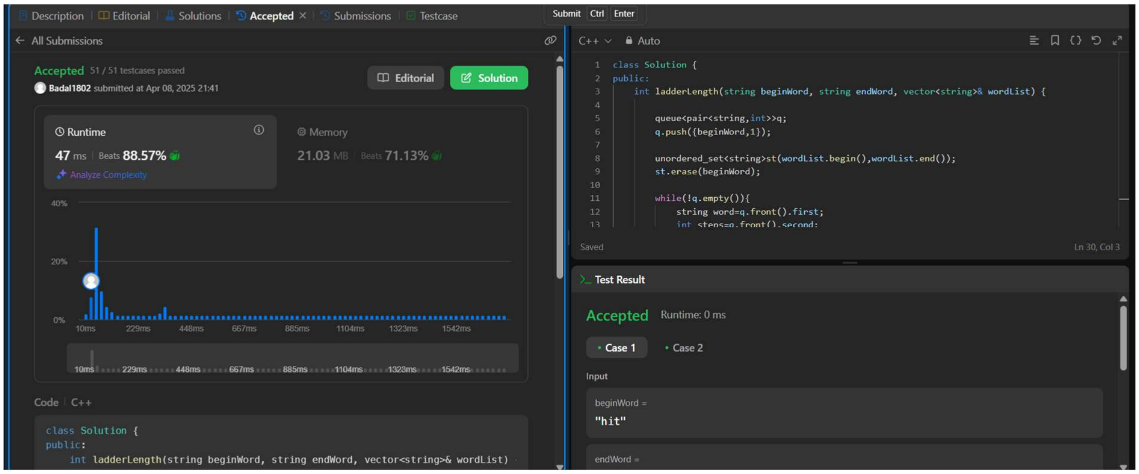
Word Ladder

CODE:

```
class Solution {
public:
  int ladderLength(string beginWord, string endWord, vector<string>& wordList) {

    queue<pair<string,int>>q;
    q.push({beginWord,1});

    unordered_set<string>st(wordList.begin(),wordList.end());
    st.erase(beginWord);

    while(!q.empty()){
      string word=q.front().first;
      int steps=q.front().second;
      q.pop();

      if(word==endWord) return steps;
      for(int i=0;i<word.size();i++){
        char original=word[i];
        for(int ch='a';ch<='z';ch++){
          word[i]=ch;
          if(st.find(word)!=st.end()){
            st.erase(word);
            q.push({word,steps+1});
          }
        }
        word[i]=original;
      }
```

```
        }return 0;
    }
};
```

OUTPUT:



Accepted 51 / 51 testcases passed
Badal1802 submitted at Apr 08, 2025 21:41

47 ms | Beats 88.57%

21.03 MB | Beats 71.13%

```
class Solution {
public:
    int ladderLength(string beginWord, string endWord, vector<string>& wordList)
```

```
class Solution {
public:
    int ladderLength(string beginWord, string endWord, vector<string>& wordList) {

        queue<pair<string,int>>q;
        q.push({beginWord,1});

        unordered_set<string>st(wordList.begin(),wordList.end());
        st.erase(beginWord);

        while(!q.empty()){
            string word=q.front().first;
            int steps=q.front().second;
```

Accepted Runtime: 0 ms

beginWord =
"hit"

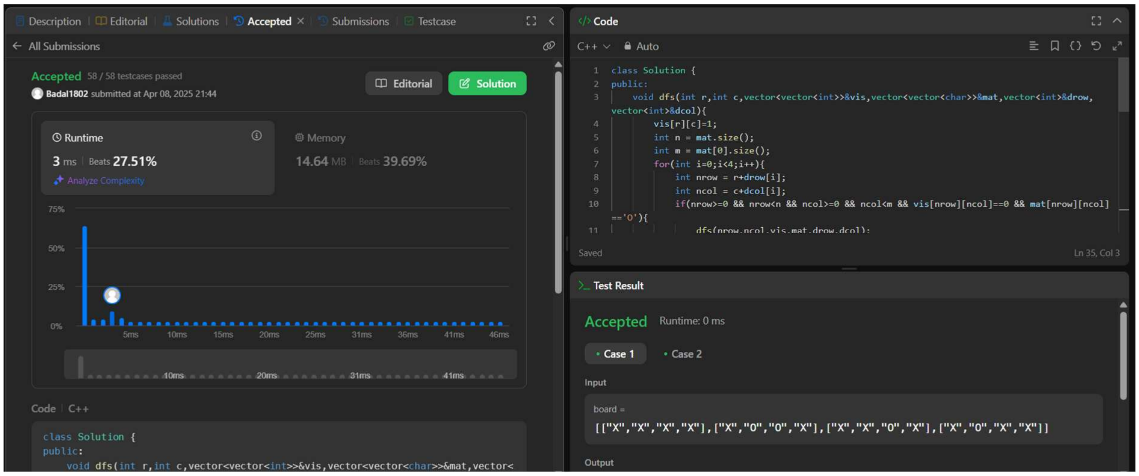
endWord =

Surrounded Regions

CODE:

```
class Solution {
public:
    void dfs(int r,int
c,vector<vector<int>>&vis,vector<vector<char>>&mat,vector<int>&drow,vector<int>&dcol){
        vis[r][c]=1;
        int n = mat.size();
        int m = mat[0].size();
        for(int i=0;i<4;i++){
            int nrow = r+drow[i];
            int ncol = c+dcol[i];
            if(nrow>=0 && nrow<n && ncol>=0 && ncol<m && vis[nrow][ncol]==0 && mat[nrow][ncol]=='O'){
                dfs(nrow,ncol,vis,mat,drow,dcol);
            }
        }
    }

    void solve(vector<vector<char>>& mat) {
        int n = mat.size();
        int m =mat[0].size();
        vector<int>drow = {-1,0,1,0};
        vector<int>dcol = {0,1,0,-1};
        vector<vector<int>>vis(n,vector<int>(m,0));
        for(int i =0;i<m;i++){
            if(vis[0][i]==0 && mat[0][i]=='O')dfs(0,i,vis,mat,drow,dcol);
            if(vis[n-1][i]==0 && mat[n-1][i]=='O')dfs(n-1,i,vis,mat,drow,dcol);
        }

        for(int  i=0;i<n;i++){
            if(vis[i][0]==0 && mat[i][0]=='O')dfs(i,0,vis,mat,drow,dcol);
            if(vis[i][m-1]==0 && mat[i][m-1]=='O')dfs(i,m-1,vis,mat,drow,dcol);
        }
        for(int i=0;i<n;i++){
            for(int j=0;j<m;j++){
                if(vis[i][j]==0 && mat[i][j]=='O')mat[i][j]='X';
            }
        }
    }
};
```

OUTPUT:



Accepted 58 / 58 testcases passed

Badal1802 submitted at Apr 08, 2025 21:44

Runtime: 3 ms | Beats 27.51%

Memory: 14.64 MB | Beats 39.69%

Code | C++

```
class Solution {
public:
    void dfs(int r,int c,vector<vector<int>>&vis,vector<vector<char>>&mat,vector<
```

```
class Solution {
public:
    void dfs(int r,int c,vector<vector<int>>&vis,vector<vector<char>>&mat,vector<int>&drow,
vector<int>&dcol){
        vis[r][c]=1;
        int n = mat.size();
        int m = mat[0].size();
        for(int i=0;i<4;i++){
            int nrow = r+drow[i];
            int ncol = c+dcol[i];
            if(nrow>=0 && nrow<n && ncol>=0 && ncol<m && vis[nrow][ncol]==0 && mat[nrow][ncol]
=='O'){
                dfs(nrow,ncol,vis,mat,drow,dcol);
```

Ln 35, Col 3

Accepted Runtime: 0 ms

Input

```
board =
[["X","X","X","X"],["X","O","O","X"],["X","X","O","X"],["X","O","X","X"]]
```

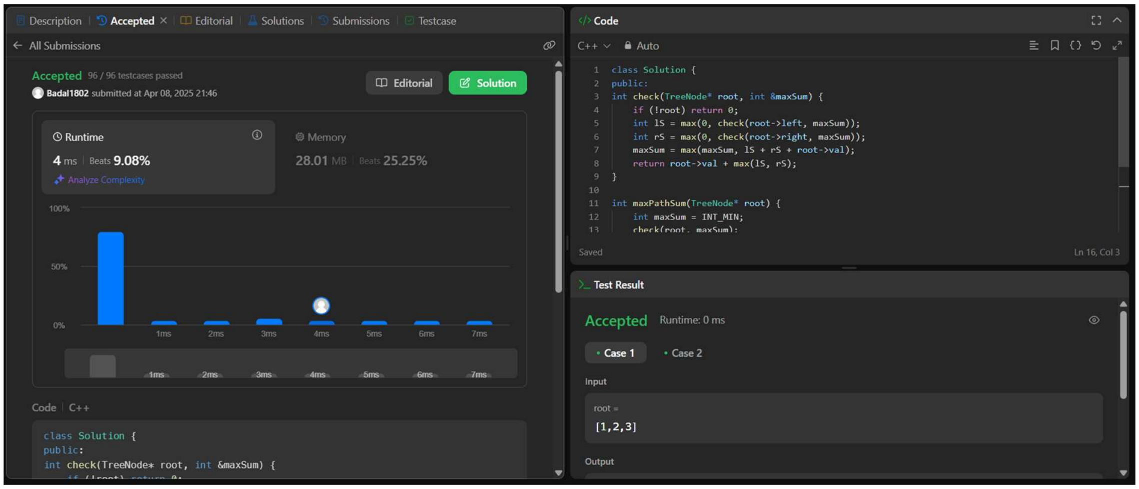
Output

Binary Tree Maximum Path Sum

CODE:

```
class Solution {

public:

int check(TreeNode* root, int &maxSum) {

   if (!root) return 0;

   int lS = max(0, check(root->left, maxSum));

   int rS = max(0, check(root->right, maxSum));

   maxSum = max(maxSum, lS + rS + root->val);

   return root->val + max(lS, rS);

}


int maxPathSum(TreeNode* root) {

   int maxSum = INT_MIN;

   check(root, maxSum);

   return maxSum;

}
};
```

OUTPUT:



Accepted 96 / 96 testcases passed

Badal1802 submitted at Apr 08, 2025 21:46

Runtime 4 ms | Beats 9.08%

Memory 28.01 MB | Beats 25.25%

Code | C++

```
class Solution {
public:
int check(TreeNode* root, int &maxSum) {
    if (!root) return 0;
```

```
class Solution {
public:
int check(TreeNode* root, int &maxSum) {
    if (!root) return 0;
    int lS = max(0, check(root->left, maxSum));
    int rS = max(0, check(root->right, maxSum));
    maxSum = max(maxSum, lS + rS + root->val);
    return root->val + max(lS, rS);
}

int maxPathSum(TreeNode* root) {
    int maxSum = INT_MIN;
    check(root, maxSum);
```

Saved

Ln 16, Col 3

Accepted Runtime: 0 ms

root =

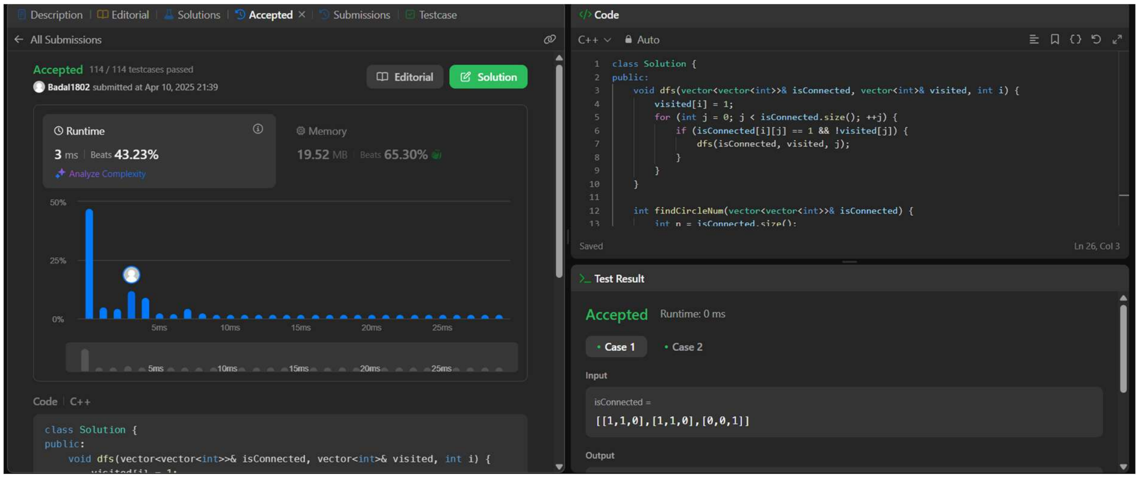
[1,2,3]

Friend Circles

CODE:

```
class Solution {
public:
    void dfs(vector<vector<int>>& isConnected, vector<int>& visited, int i) {
        visited[i] = 1;
        for (int j = 0; j < isConnected.size(); ++j) {
            if (isConnected[i][j] == 1 && !visited[j]) {
                dfs(isConnected, visited, j);
            }
        }
    }

    int findCircleNum(vector<vector<int>>& isConnected) {
        int n = isConnected.size();
        vector<int> visited(n, 0);
        int count = 0;

        for (int i = 0; i < n; ++i) {
            if (!visited[i]) {
                dfs(isConnected, visited, i);
                count++;
            }
        }

        return count;
    }
};
```

OUTPUT:



Accepted 114 / 114 testcases passed

Badal1802 submitted at Apr 10, 2025 21:39

Runtime 3 ms | Beats 43.23%

Memory 19.52 MB | Beats 65.30%

Code | C++

```
class Solution {
public:
    void dfs(vector<vector<int>>& isConnected, vector<int>& visited, int i) {
        visited[i] = 1;
```

```
class Solution {
public:
    void dfs(vector<vector<int>>& isConnected, vector<int>& visited, int i) {
        visited[i] = 1;
        for (int j = 0; j < isConnected.size(); ++j) {
            if (isConnected[i][j] == 1 && !visited[j]) {
                dfs(isConnected, visited, j);
            }
        }
    }

    int findCircleNum(vector<vector<int>>& isConnected) {
        int n = isConnected.size();
```

Saved

Ln 26, Col 3

Accepted Runtime: 0 ms

Input

isConnected =

[[1,1,0],[1,1,0],[0,0,1]]

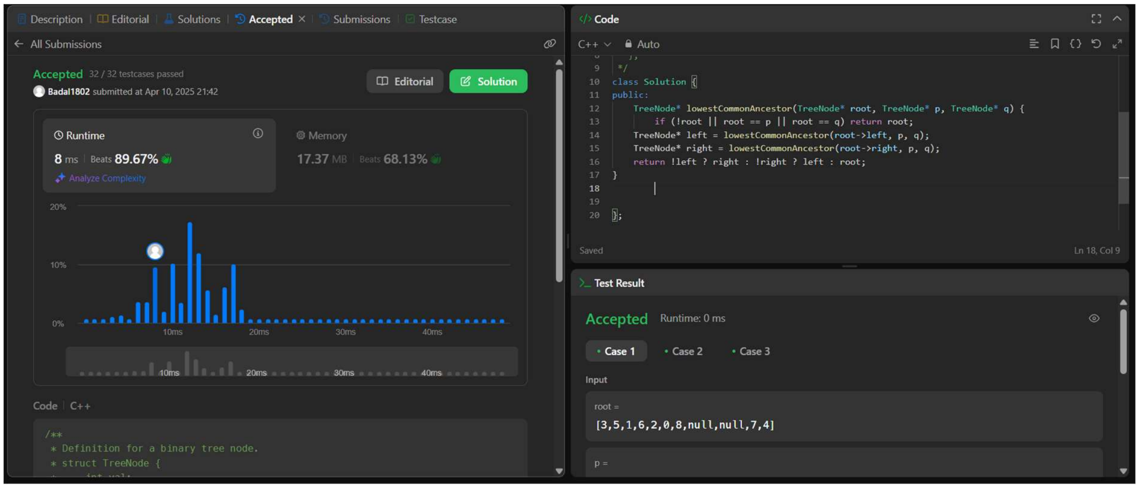
Output

Lowest Common Ancestor of a Binary Tree

CODE:

```
class Solution {
public:
   TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
      if (!root || root == p || root == q) return root;
   TreeNode* left = lowestCommonAncestor(root->left, p, q);
   TreeNode* right = lowestCommonAncestor(root->right, p, q);
   return !left ? right : !right ? left : root;
}
};
```

OUTPUT:



Accepted 32 / 32 testcases passed

Badal1802 submitted at Apr 10, 2025 21:42

Runtime: 8 ms | Beats 89.67%

Memory: 17.37 MB | Beats 68.13%

Code | C++

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
```

```
 */
class Solution {
public:
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
        if (!root || root == p || root == q) return root;
    TreeNode* left = lowestCommonAncestor(root->left, p, q);
    TreeNode* right = lowestCommonAncestor(root->right, p, q);
    return !left ? right : !right ? left : root;
}


};
```

Saved

Ln 18, Col 9

Accepted Runtime: 0 ms

Input

root =

```
[3,5,1,6,2,0,8,null,null,7,4]
```

p =

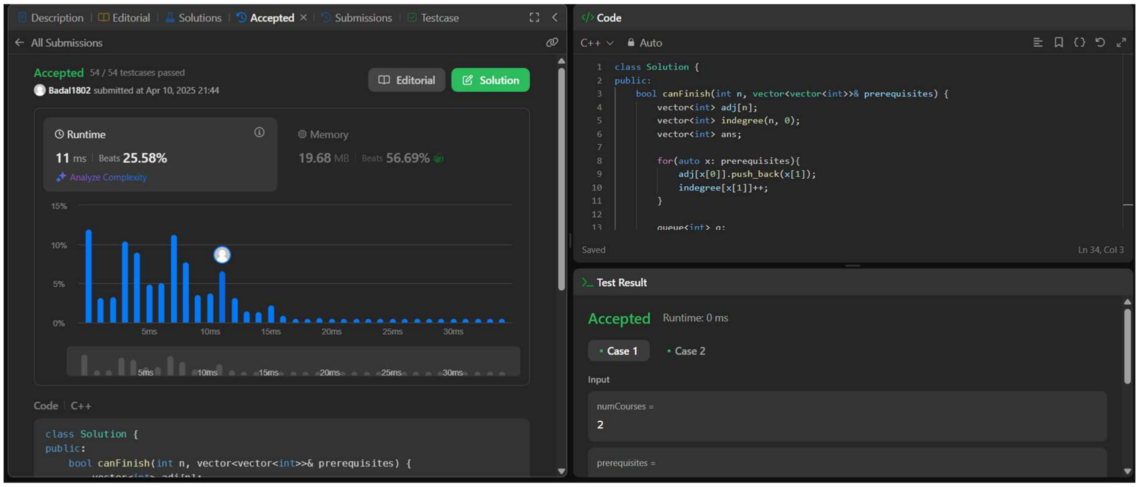
Course Schedule

CODE:

```
class Solution {
```

```
public:
    bool canFinish(int n, vector<vector<int>>& prerequisites) {
        vector<int> adj[n];
        vector<int> indegree(n, 0);
        vector<int> ans;

        for(auto x: prerequisites){
            adj[x[0]].push_back(x[1]);
            indegree[x[1]]++;
        }

        queue<int> q;
        for(int i = 0; i < n; i++){
            if(indegree[i] == 0){
                q.push(i);
            }
        }

        while(!q.empty()){
            auto t = q.front();
            ans.push_back(t);
            q.pop();

            for(auto x: adj[t]){
                indegree[x]--;
                if(indegree[x] == 0){
                    q.push(x);
                }
            }
        }
        return ans.size() == n;
    }
};
```

OUTPUT:



Accepted 54 / 54 testcases passed

Badal1802 submitted at Apr 10, 2025 21:44

Runtime

11 ms | Beats 25.58%

Analyze Complexity

Memory

19.68 MB | Beats 56.69%

Code | C++

```
class Solution {
public:
    bool canFinish(int n, vector<vector<int>>& prerequisites) {
```

```
class Solution {
public:
    bool canFinish(int n, vector<vector<int>>& prerequisites) {
        vector<int> adj[n];
        vector<int> indegree(n, 0);
        vector<int> ans;

        for(auto x: prerequisites){
            adj[x[0]].push_back(x[1]);
            indegree[x[1]]++;
        }

        queue<int> q;
```

Saved

Ln 34, Col 3

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

numCourses =

2

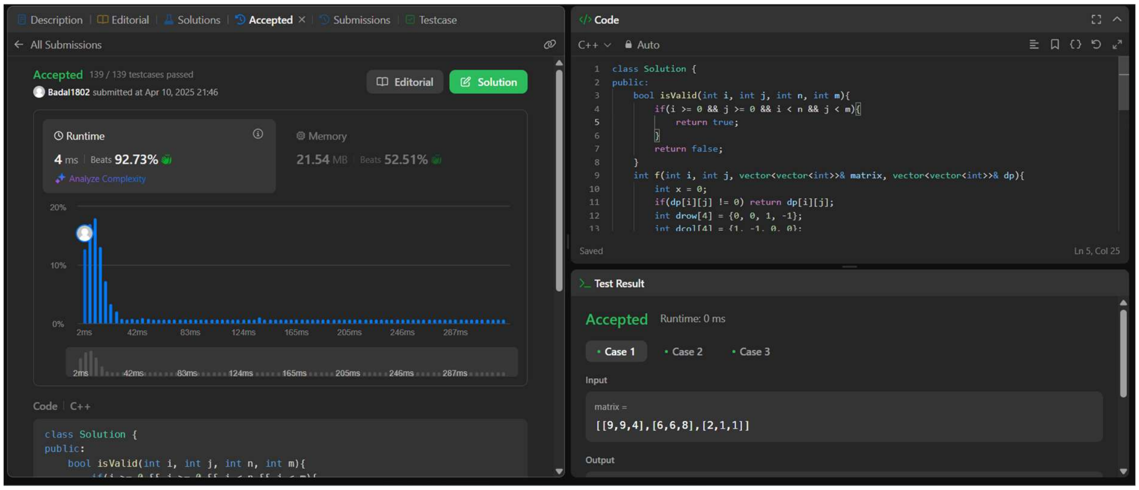
prerequisites =

Longest Increasing Path in a Matrix

CODE:

```
class Solution {
public:
    bool isValid(int i, int j, int n, int m){
        if(i >= 0 && j >= 0 && i < n && j < m){
            return true;
        }
        return false;
    }
    int f(int i, int j, vector<vector<int>>& matrix, vector<vector<int>>& dp){
        int x = 0;

        if(dp[i][j] != 0) return dp[i][j];

        int drow[4] = {0, 0, 1, -1};

        int dcol[4] = {1, -1, 0, 0};

        for(int k = 0; k <4; k++){

            int nrow = i+drow[k];

            int ncol = j+dcol[k];

            if(isValid(nrow, ncol, matrix.size(), matrix[0].size()) && matrix[nrow][ncol] >
matrix[i][j]){

                x = max(x, f(nrow, ncol, matrix, dp));

            }

        }

        return dp[i][j] = x+1;

    }

    int longestIncreasingPath(vector<vector<int>>& matrix) {

        int n = matrix.size();

        int m = matrix[0].size();

        vector<vector<int>>dp(n, vector<int>(m, 0));

        int ans = 0;

        for(int i = 0; i < n; i++){

            for(int j = 0; j < m; j++){

                ans = max(ans, f(i, j, matrix, dp));

            }

        }

        return ans;

    }

};
```

OUTPUT:



Accepted 139 / 139 testcases passed

Badal1802 submitted at Apr 10, 2025 21:46

Runtime: 4 ms | Beats 92.73%

Memory: 21.54 MB | Beats 52.51%

Code | C++

```
class Solution {
public:
    bool isValid(int i, int j, int n, int m){
```

```
class Solution {
public:
    bool isValid(int i, int j, int n, int m){
        if(i >= 0 && j >= 0 && i < n && j < m){
            return true;
        }
        return false;
    }
    int f(int i, int j, vector<vector<int>>& matrix, vector<vector<int>>& dp){
        int x = 0;
        if(dp[i][j] != 0) return dp[i][j];
        int drow[4] = {0, 0, 1, -1};
        int dcol[4] = {1, -1, 0, 0};
```

Saved

Ln 5, Col 25

Accepted Runtime: 0 ms

matrix =

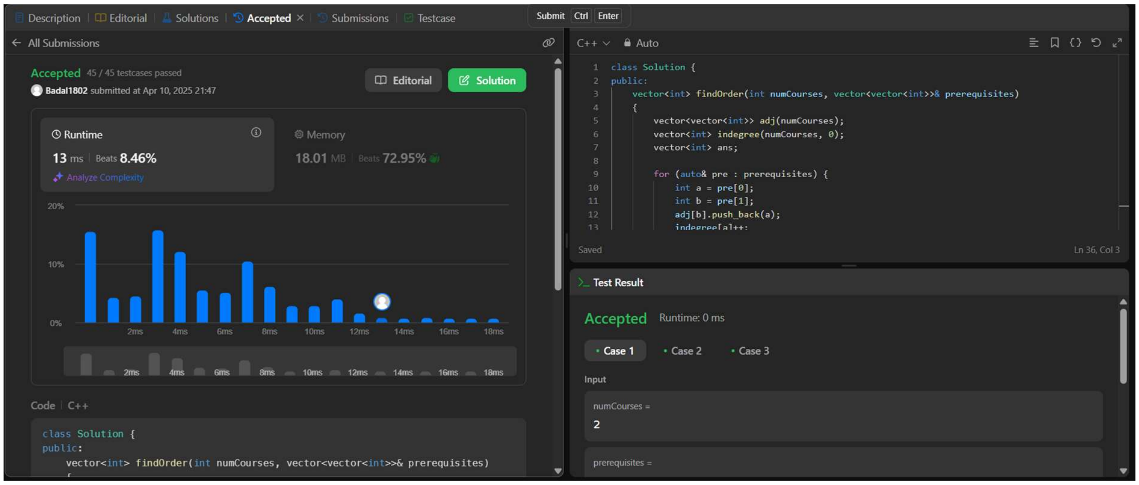
[[9,9,4],[6,6,8],[2,1,1]]

Course Schedule II

CODE:

```
class Solution {
public:
    vector<int> findOrder(int numCourses, vector<vector<int>>& prerequisites)
    {
        vector<vector<int>> adj(numCourses);
        vector<int> indegree(numCourses, 0);
        vector<int> ans;

        for (auto& pre : prerequisites) {
            int a = pre[0];
            int b = pre[1];
            adj[b].push_back(a);
            indegree[a]++;
        }
        queue<int> q;
        for (int i = 0; i < numCourses; i++) {
            if (indegree[i] == 0)
                q.push(i);
        }
        while (!q.empty()) {
            int node = q.front(); q.pop();
            ans.push_back(node);
            for (int neighbor : adj[node]) {
                indegree[neighbor]--;
                if (indegree[neighbor] == 0)
                    q.push(neighbor);
            }
        }
        if(ans.size() != numCourses) return {};
        return ans;
    }
};
```

OUTPUT:



Accepted 45 / 45 testcases passed

Badal1802 submitted at Apr 10, 2025 21:47

Runtime

13 ms | Beats 8.46%

Memory

18.01 MB | Beats 72.95%

Code | C++

```
class Solution {
public:
    vector<int> findOrder(int numCourses, vector<vector<int>>& prerequisites)
```

```
class Solution {
public:
    vector<int> findOrder(int numCourses, vector<vector<int>>& prerequisites)
    {
        vector<vector<int>> adj(numCourses);
        vector<int> indegree(numCourses, 0);
        vector<int> ans;

        for (auto& pre : prerequisites) {
            int a = pre[0];
            int b = pre[1];
            adj[b].push_back(a);
            indegree[a]++;
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

numCourses =

2

prerequisites =